Design and Analysis of Algorithms Assignment 2

Harrison Lee, Alex Zhao

January 25, 2019

Question 1. *(4.9) Greedy Algorithm*

Claim: Given two nodes, s and t, in n-node undirected graph $G = (V, E)$ with a distance greater than $n/2$, there exists some node v not equal to either s or t that, when deleted, destroys all paths from s to t.

Proof. A path of distance greater than $n/2$ takes $n/2 + 1$ nodes at least. Excluding s and t this is $(n-2)/2 + 1 = n/2$ nodes. Let's call this path "path A".

For node v to be deletable without destroying the other path, path B, v cannot be in B.

B must also have a distance greater than $n/2$ to maintain s and ts' distance.

B cannot share nodes with A other than s and t and requires $n/2$ nodes unique from A.

There are only $n - 2$ non-s or non-t nodes, but A and B require a total of n unique nodes, so B cannot exist.

$\square$

Algorithm 1. *Find node v*

Begin with a Depth-First-Search, as written in the textbook. Use it to find the shortest path from s to t.

Mark all nodes discovered in that shortest path as "Used", numbering them by their distance to t.

Repeat Depth-First-Search, starting from s, but do not traverse past "Used" nodes. Instead, mark them as "Re-Found". Save whichever "Re-Found" node that is closest to t.

Once Depth-First-Search fails and cannot continue, return the saved "Re-Found" node as v.

Proof. Prove this algorithm is $O(n + m)$:

Depth-First-Search is known to be $O(n + m)$, as noted in the textbook. Each edge and node is traversed at most once.

This algorithm conducts Depth-First-Search twice.

This algorithm is $O(2n + 2m)$, which is close enough to $O(n + m)$ for our purposes.

$\square$

Question 2. *(6.11) Dynamic Programming*

Given: You have n weeks and s_i parts to be produced each of those weeks. You must decide between two shipping companies, Company A which charges $r * s$ to ship in a given week while Company B charges c each week in blocks of four consecutive weeks.

Find: A schedule deciding between company A or B for each of those n weeks while following company B's restrictions. Cost is the amount paid in shipping costs.

Give a polynomial time algorithm that takes a sequence of supply values, $s_1, s_2, \ldots, s_n$ and returns a schedule of minimum cost.

Algorithm 2. *Find the optimal schedule.*

Proof. Subproblems: $OPT(j)$, or the optimal schedule from week 0 to week j. This can be expanded from week 4 to week n.

Recurrence: $OPT(j) = min\{ \begin{matrix} r * s_j + OPT(j-1) \\ 4 * c + OPT(j-4) \end{matrix}$, where $OPT(j)$ represents the optimal shipping costs possible from week 0 to week j. $r, s,$ and c are as defined in the problem, referring to company A's cost per weight unit, total weight during week j, and company B's cost per week respectively.

Full Algorithm:

OPT[0, 1, 2, 3] = [0, $s_1 * r, s_1 * r + s_2 * r, s_1 * r + s_2 * r + s_3 * r$]

for j = 0, j $\leq$ n, j++

  if $OPT[j-4] + 4 * c < OPT[j-1] + r * s_j$

    $OPT[j] = OPT[j-4] + 4 * c$

  else:

    $OPT[j] = OPT[j-1] + r * s_j$

$returnOPT[n]$

Running Time: $O(n)$. Each week is traversed once, while two already calculated $OPT(j)$s are accessed, $OPT(j-1)$ and $OPT(j-4)$, each week.

□